Preprint
UCRL-JC-141577

An Adaptive Signed Distance Transform for Curves with Guaranteed Error Bounds

D. E. Laney, M. A. Duchaineau, N. L. Max

This article was submitted to
International Electrical and Electronic Engineering Technical Committee on Visualization and Graphics - Syjmposium on Visualization
Ascona, Switzerland
May 28-30, 2001

U.S. Department of Energy

Lawrence Livermore National Laboratory

December 4, 2000

An Adaptive Signed Distance Transform for Curves with Guaranteed Error Bounds

Daniel E. Laney[1], Mark A. Duchaineau[2], Nelson L. Max[1,2]

[1] Department of Applied Science, University of California at Davis
[2] Lawrence Livermore National Laboratory* * *

Abstract. We present an adaptive signed distance transform algorithm for curves in the plane. The algorithm provides guaranteed error bounds with a selective refinement approach. The domain over which the signed distance function is desired is adaptive triangulated and piecewise discontinuous linear approximations are constructed within each triangle. The resulting transform performs work only were requested and does not rely on a preset sampling rate or other constraints.

1 Introduction

In this paper we present an adaptive signed distance approximation for curves in two dimensions. Our transform produces an approximation of the signed distance function of a given curve. A signed distance function defines a scaler field that specifies the minimum distance to a curve for every point in the plane, with the sign distinguishing between inside and outside.

Distance functions have been used in image processing for some time. Distance functions in three dimensions are also a promising shape representation with interesting applications in geometric design and surface reconstruction. Furthermore, they are well suited to representing dynamic curves and surfaces with changing topology. However, most research relies on distance transforms which sample a distance function without regard to sampling rate requirements. In addition, most transform algorithms for surfaces do not provide error bounds.

Our goal is an adaptive distance transform which provides guaranteed error bounds and enables local refinement operations to increase accuracy. The algorithm should not require preset sampling rates or other constraints. We are investigating distance functions in the plane as a precursor to a full three dimensional method. In two dimensions, the error analysis is simplified, and the behavior of the algorithm and data structures can be clearly visualized.

2 Overview

The distance function of a curve defines a scaler field that specifies the minimum distance to the curve at every point in space. In addition, the distance may be signed in

* * * {laney1, duchaineau1, max2}@llnl.gov

order to differentiate between inside and outside. We formally define the signed distance function of a curve or set of curves C as follows:

$$\mathrm{d}(\boldsymbol{x}) := \mathrm{sign}(\boldsymbol{x}) * \min_{\forall \boldsymbol{y} \in C}(\| \boldsymbol{x} - \boldsymbol{y} \|) \tag{1}$$

where C is the set of points on the input curve(s), $\boldsymbol{x}$ and $\boldsymbol{y}$ are position vectors, $\boldsymbol{x}, \boldsymbol{y} \in \Re^2$, $\boldsymbol{y} \in C$, and $\mathrm{sign}(\boldsymbol{x})$ returns negative if $\boldsymbol{x}$ inside the curve, and positive if outside.

The algorithm presented in this paper creates an approximation $\tilde{d}(\boldsymbol{x})$ of the signed distance function over a domain D for a set of input curves C. The input curves C must have the following properties within D:

1. The curve(s) must be C^0 continuous everywhere and C^1 continuous everywhere except at a finite number of points.
2. Curve end points may only occur on the boundary of D.
3. The curves must partition the domain D into an inside and outside labeled by negative and positive distances.

The algorithm creates an adaptive triangulation of the domain D. A linear approximation of the signed distance function is computed within each triangle with respect to the subset of the input curve C which may affect the distance function within the triangle. The resulting piecewise linear approximation is discontinuous at the vertices and edges of the triangle mesh. The error bounds of the distance approximation are guaranteed, and therefore the error intervals will overlap at the vertices and edges of the distance mesh. A continuous version of the distance approximation can be extracted that satisfies these error bounds at all points.

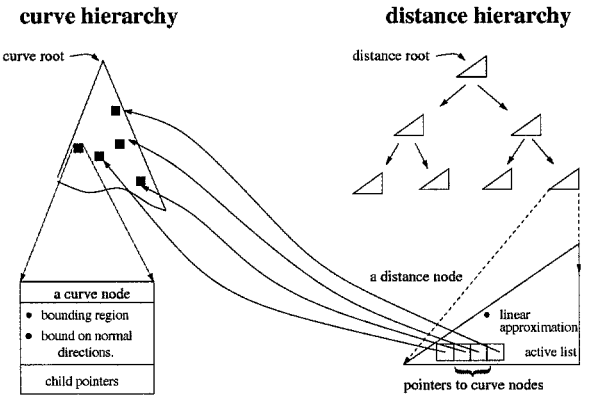


Fig. 1. The two hierarchies of the distance transform algorithm.

The distance transform operates on two hierarchies. The curve hierarchy is the input to the transform and consists of bounding boxes on the input curves. The distance hierarchy is the output of the algorithm and is a triangle bintree on the domain D of the distance function (see section 5). Figure 1 shows the relation between these hierarchies.

The left half of figure 1 schematically depicts the curve hierarchy. The filled black boxes represent individual nodes of the curve hierarchy. Below the curve hierarchy is the layout of a typical node. Each node provides a bounding region that contains the curve subset it represents and a bound on the normal directions of the curve within the bounding region. The bounding region allows fast culling of larger subsets of the curve. The bound on the curve normal direction enables the algorithm to establish tighter approximations when the curve is nearly linear.

The right half of figure 1 depicts the distance hierarchy. The distance hierarchy consists of a triangle bintree that defines a piecewise discontinuous linear approximation of the distance function. Each node maintains an active list of curve hierarchy nodes as shown at the bottom right of figure 1. The active list is culled such that it contains the subset of the curve which may contribute to the distance function within a node.

Fig. 2. An isocontour bounding box hierarchy.

For this paper we have chosen to develop the algorithm using isocontours of a regularly sampled scaler field. This allows several sets of curves to be generated, and we can see the behavior of the algorithm on actual simulation data. The curve properties presented above are obtained by using a bilinear interpolation of the scaler field samples. The isocontour example differs from a parametric curve representation in that the actual isocontour is not extracted from the scaler field. Instead, the gradient information of the field is used to construct a linear distance approximation. In this case we are transforming from one scaler field representation to another. It is expected that for some data it may not be possible or necessary to first extract an isocontour and then compute a distance field. For instance, our algorithm could be placed between the raw scaler field data and a progressive isocontouring system such as [7].

For the isocontour example in this paper, we first preprocess the original scaler field by building a minmax quadtree [9]. Minmax quadtrees store the minimum and maximum values of the scaler field contained within each node. Figure 2a shows the scaler field we will be using in the description of our algorithm in the following sections. A bounding box hierarchy is selected by specifying an isovalue. All nodes of the scaler field quadtree which contain that isovalue form the bounding box hierarchy. Figures 2b-d show three levels of the bounding box hierarchy for a specific isocontour.

Figure 3 shows an overview of the distance transform algorithm. The distance transform proceeds by adaptively refining the triangle bintree as shown in figure 3a. Figure 3b shows the active list of isocontour bounding boxes maintained by the red triangle of the bintree. The active list is culled so that only the subset of the isocontour which potentially contributes to the distance function remains. A linear approximation of the distance function is computed within each triangle of the distance bintree. Figure 3c

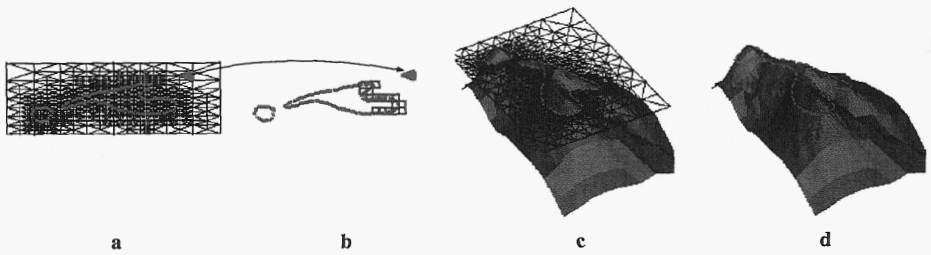


Fig. 3. Adaptive refinement of the distance approximation.

shows a continuous version of the discontinuous piecewise linear approximation of the signed distance function. The cleft in figure 2b is in the foreground. The approximate isocontour (shown in red) is extracted by taking the zero set of the continuous version of the distance approximation. The triangle bintree is drawn at the zero set level to show that positive distance values occur inside the isocontour loop. Figure 3d shows the same distance approximation with the triangle bintree omitted.

3 Related Work

Distance functions and distance transforms have been used in surface reconstruction and modeling research for some time. In [5] a best fit plane was computed for each point in the input dataset based on k neighboring points. These neighboring points form a k-neighborhood. Consistent orientations of the best fit planes is accomplished by traversing a Riemannian Graph which connects each point to the others in its k-neighborhood and vice versa. A continuous distance approximation is obtained by sampling the distances from the best fit planes to the vertices of a regular subdivision of space. A propagation algorithm is used to extract the surface from the resulting sampled scaler field. The approach in this paper is similar due to the fact that it is the curve normal information that guide the linear approximation.

In [1] α-shapes [3] are used to approximate the signed distance transform for data from 3D scans. In a preprocessing phase the Delaunay triangulation of the sample points is computed. Next, a voronoi diagram and a family of alpha shapes are constructed. The α-shapes are used to provide a "good" linear approximation to the surface to be reconstructed for each tetrahedra in the triangulation. To estimate the distance at a query point, the containing tetrahedron is used to determine the sign and the voronoi diagram is used to select a surface sample point with which to measure the distance. The method uses the resulting distance approximation to adaptively approximate the surface and a scaler field using trivariate Bernstein-Bezier functions defined over tetrahedra.

Another popular method for obtaining approximate signed distance functions is the closest point propagation technique. In [2] a modified fast marching method [8] was used in the following way: A set of sample points on the surface to be approximated is generated. Next, the points are embedded in a volumetric grid is and all cells containing

surface points are initialized by the closest surface point to each cell vertex. The closest point information is propagated from the thin shell of grid vertices to their neighbors using a priority queue. At each iteration the grid vertex with the smallest distance is removed from the queue and its closest point information is passed to its neighbors. The neighbors replace their current closest point if the distance to the closest surface point of the just frozen vertex is smaller. This heuristic is fast an works well in most cases. One disadvantage is that the grid resolution is fixed at the start of the algorithm.

An adaptive sampling approach [4] was presented which attempts to alleviate data storage issues which occur when sampled distance volumes are used. In there approach it was assumed a distance function was available. A top down method was used to sample the distance function at the vertices of an octree. The adaptively sampled distance function was then rendered and edited. Whenever more detail was required the octree was refined further and more samples computed from the distance function.

4 The Curve Hierarchy

In this section we describe the the curve hierarchy required as input to the distance transform. We begin by stating the properties of the input curve hierarchy for the general case, then describe in detail the isocontour hierarchy used in this paper.

4.1 Curve Hierarchy Requirements

Let C denote the set of points on the input curve. Let each node of the curve hierarchy have a unique index α. The distance transform algorithm requires a hierarchy on the input curve with the following properties:

1. Each node α has a bounding region B_α in the xy plane which contains a subset of the curve.
2. Each node α has a bound on the directions of all unit normals of the curve contained in the node. The bound is represented by a normal wedge $(\boldsymbol{n}_\alpha, \psi_\alpha)$ with central unit normal $\boldsymbol{n}_\alpha$ and opening angle $0 \leq \psi_\alpha \leq \pi$.

A unit vector $\boldsymbol{v}$ is contained in a normal wedge for node α if the following condition is met:

$$\boldsymbol{v} \cdot \boldsymbol{n}_\alpha \geq \cos(\psi_\alpha) \tag{2}$$

As stated previously, the curve may not be C^1 at all points. In cases of a discontinuity of the curve normals, the direction bound must be chosen such that equation (2) is satisfied for any normal in the bounding region.

Any number of data structures and curve definitions could be used to construct a curve hierarchy. One potential application area for an adaptive distance transform is in the visualization of isosurfaces arising from scientific simulations. Therefore, we chose to define a curve hierarchy in terms of scaler field data generated by simulation codes. The next subsection describes this isocontour hierarchy.

4.2 Constructing An Isocontour Hierarchy

Given a regularly sampled scaler field, we assume a rectilinear mesh with the samples at the vertices. The resulting piecewise bilinear field produces isocontours which are C^0 everywhere, but not C^1 continuous at the vertices and edges of the rectilinear mesh. These isocontours provide the minimal set of properties stated in section 2.

We construct a quadtree on the scaler field. Each node of the quadtree stores bounds on the scaler field value as well as bounds on the gradient components. A bounding box hierarchy for a particular isocontour is implied in the quadtree data structure when nodes which do not contain the isovalue are ignored. In this way, a hierarchy on the scaler field contains hierarchies for all isocontours in the field.

Bounds on the normal directions of the isocontours in a quadtree node follow from the gradients of the scaler field. Given a point where the scaler field is C^1 continuous, the normal of the isocontour passing through the point is given by the gradient of the scaler field at that point. Gradient discontinuities at vertices and edges of leaf nodes give rise to different normals depending on which direction the limit normal is sought. Taking these into consideration, a loose bound on the normal directions of an isocontour can be obtained by bounding the scaler field gradients within a quadtree node.

Figure 4 shows an example scaler field and three levels of the quadtree hierarchy for a particular isovalue. The gradient bounds are drawn inside each node with the opening angles denoted by dotted arcs. The isocontour is shown for reference.

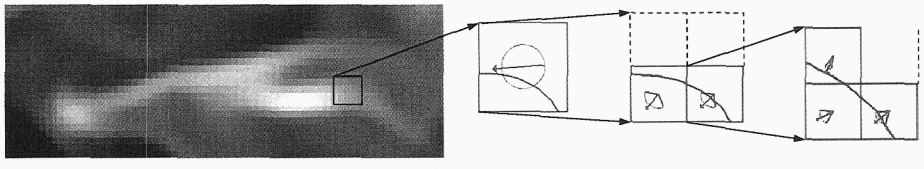


Fig. 4. (Top) A sampled scaler field rendered with one pixel/sample. A single quadtree node is highlighted. (Bottom) Three levels of the isocontour bounding box hierarchy.

For the scaler fields in this paper a branch on need quadtree was constructed [9]. Branch on need quadtrees can be used with datasets whose dimensions are not powers of two. Figure 5 shows four levels of a branch on need quadtree. The construction begins by computing the scaler field and gradient bounds for the leaf nodes. These bounds are propagated towards the root of the tree by merging the bounds of the children at each parent.

Bounds on the components of the scaler field gradient within a leaf node are computed by differentiating the basis functions in parametric (uv) coordinates and taking the minimum and maximum values of each component. Figure 6 shows a unit square in parametric coordinates with the bilinear basis functions. If $f_{0\ldots3}$ are the scaler field values corresponding to the vertices $0\ldots3$ in parametric space, then the scaler field is interpolated as $F(u,v) = \sum_{j=0}^{3} f_j N_j(u,v)$. We will label the components of the gradient as F_u and F_v.

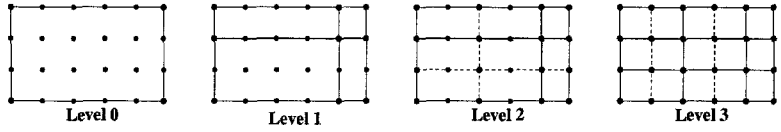
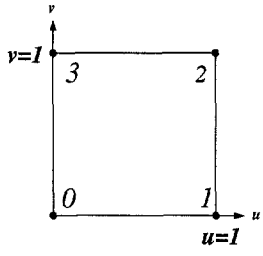


Fig. 5. A branch on need quadtree splits each node such that along each edge the number of vertices plus one is a power of two.



$$N_0(u,v) = (1-u)(1-v)$$
$$N_1(u,v) = u(1-v)$$
$$N_2(u,v) = uv$$
$$N_3(u,v) = (1-u)v$$

Fig. 6. A unit square in parameter space with the vertices ordered as shown.

The gradient component bounds are computed and transformed from uv space to xy space. The gradient component bounds of child nodes are merged in a parent node by taking the minimum of the minimums and maximum of the maximums of each component. These bounds contain all possible gradients, including all possible gradients at edges and vertices where the scaler field is discontinuous. Finally, the gradient bounds are converted to a normal vector wedge as shown in figures 7a and 7b. After the two vectors in 7b are normalized a normal wedge is found by finding the bisecting unit vector and computing an opening angle. Note that this is only valid if the origin is not contained inside the gradient bounding box; otherwise the opening angle is set to π.

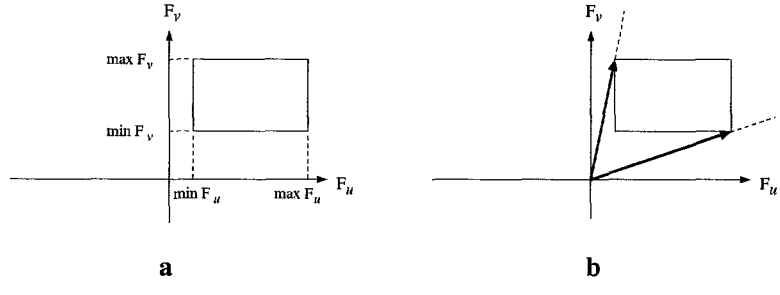


Fig. 7. (a) Gradient bounds plotted as a box in gradient component space. (b) Converting a gradient bound into a normal wedge.

The scaler field hierarchy has loose bounds on the normal direction because the direction bounds are valid for all isocontours within a particular node. This forces the distance transform algorithm to subdivide more often in areas where the original scaler field gradients are widely varying. This hierarchy is similar to the data structure used in [7] in that it involves minimal preprocessing and grants access to all of the isocontours in the scaler field.

5 The Distance Hierarchy

The distance transform constructs a piecewise discontinuous linear approximation by selective refinement of a triangle bintree. In the remainder of the paper we will assume that the nodes of the distance hierarchy are indexable. We will use T_k to refer to the triangle associated with a distance node k. All other quantities associated with a node shall be denoted by subscripting it with its index. We begin this section by briefly describing triangle bintrees.

5.1 Triangle Bintrees

Figure 8a shows a few levels of a triangle bintree hierarchy. Levels zero and one of figure 8a depict the basic split operation which generates the hierarchy. The tree begins with a right isosceles triangle. The triangle is split by inserting a vertex at the midpoint of the hypotenuse and connecting it to the apex of the triangle. This operation is then applied recursively to the children. A triangle which splits creates a new edge vertex. This creates a crack problem because the triangle sharing the edge may not be split. The crack problem is solved by forcing splits until there are no vertices on any edges. Figure 8b depicts the chain of triangle splits generated when triangle T (upper left) is refined.

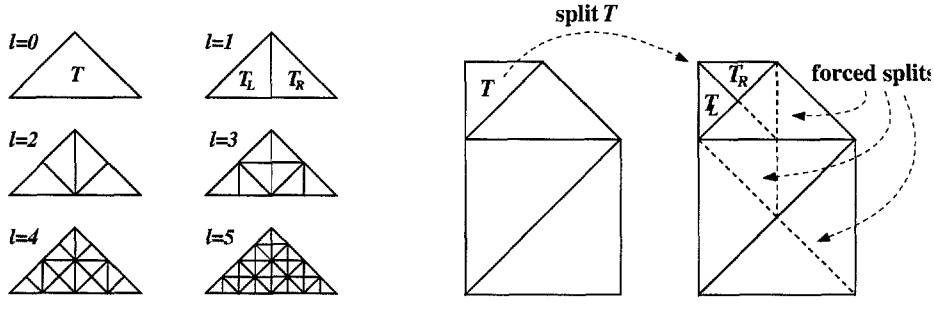


Fig. 8. The first six levels in a triangle bintree.

5.2 Linear Approximation

Each node of the distance hierarchy contains a linear approximation of the distance function within that node and an active list of curve hierarchy nodes, which is used to compute a linear approximation. In the remainder of the paper we will denote the active list of a distance node by A_k and define it as a list of indices of curve nodes as follows:

$$A_k := \alpha_0, \alpha_1, \ldots, \alpha_N \tag{3}$$

We will define a linear approximation for a node k as:

$$\tilde{d}_k(\boldsymbol{x}) = \boldsymbol{g_k} \cdot \boldsymbol{x} + c_k \tag{4}$$

In addition, an error bound on the distance function within a distance node k must satisfy:

$$\max_{\forall \boldsymbol{x} \in T_k}(\mathrm{d}(\boldsymbol{x})| - |\tilde{d}_k(\boldsymbol{x})|) \leq \varepsilon_k \tag{5}$$

where the gradient $||\boldsymbol{g_k}|| = 1$. In general, we want $\varepsilon_k < \mathrm{uerror}(k)$, where $\mathrm{uerror}(k)$ is a user defined error which depends k and may depend on other parameters as well. When the distance function is too complicated to be linearly approximated, we fall back on a constant approximation ($\boldsymbol{g_k} = 0$).

6 The Distance Transform

In this section we describe the algorithm which refine the distance hierarchy and produce linear approximations of the distance function. We will begin by outlining the top level routines in the transform algorithm. The remainder of this section will detail the bound culling routine and the linear approximation methodology.

Given a distance node k, the transform algorithm recursively subdivides k until a user defined error criterion is met over the original domain T_k. The subdivision may cause forced splittings of some triangles outside of k as mentioned in section 5.1. We assume that k contains an active list A_k of all curve nodes which may contribute to the distance function within T_k. We will formalize the notion of a contributing curve node in section 6.3. The algorithm proceeds as follows:

1. **Linear Approximation:** Compute a linear approximation of the distance function over T_k based on the active list nodes A_k. Test the active list to see if it is possible to compute a guaranteed error bound. If it is possible, compute the error and store in node k.
2. **Constant Approximation:** If no linear approximation with guaranteed error was computed, or the error of the linear approximation violates the user supplied error criterion, then compute a constant approximation as follows: Compute conservative bounds on the distance function within k with respect to the bounding regions of the curve nodes in A_k. Repeatedly refine the curve node in A_k with the largest bounding region and update the distance bounds. As the distance bounds tighten, *Bound Cull* any curve node in A_k that is unable to contribute to the distance function within k. When a curve node is *Bound Culled* label all adjacent curve nodes

as *Gap Nodes*. Stop curve node refinement when the conservative bounds produce a constant approximation which satisfies the error criterion or the bounding region of the largest node in A_k falls below a certain size.

3. **Recurse or End**: If the resulting approximation error does not satisfy the user defined error criterion, split distance node k and copy the active list A_k to both children. Repeat with step 1 for each child of k.

6.1 Computing Linear Approximations

In step 1 the process of computing a linear approximation and obtaining a guaranteed bound are decoupled from one another. This is due to the fact that curves with complex foldings, disconnected components, and gaps created during bound culling produce distance fields which are difficult to bound correctly. The error computation involves not only the configuration of the curve but also the position of the distance node. Accordingly, we describe the linear approximation techniques first, and then detail the process of obtaining a guaranteed bound in the next section.

We now outline two approaches for obtaining a linear approximation. In the first approach, the approximate distance function defined in (4) is approximated directly. For instance, the distance could be sampled over triangle T_k and $\tilde{d}_k(\boldsymbol{x})$ could be computed using a least squares method. The samples would come from points on the curve contained in the active list A_k. A second approach is to compute $\tilde{d}_k(\boldsymbol{x})$ so that its zero set approximates the contour subset contained in A_k. We use the second method in this paper and compute the linear approximation as follows.

Equation (4) implies that the zero set of a linear approximation is the equation of a line with normal $\boldsymbol{g}_k$. We set $\boldsymbol{g}_k$ to the normalized average of all central normals of curve nodes in the active list A_k. The constant c_k is obtained by choosing a point $\boldsymbol{p}$ on the curve contained in the active list A_k and solving for the zero set:

$$\boldsymbol{g}_k \cdot \boldsymbol{p} + c_k = 0 \tag{6}$$

The point $\boldsymbol{p}$ is computed by choosing an isocontour node in the active list and finding a contour intersection with one of its edges using linear interpolation. Our approach is simple and produces good results for nearly linear curves.

Figure 9 shows two approximations computed using this method. The isocontour is shown in green, the active list in blue, and the approximation in red. The distance triangle is shown for each approximation. Figure 9b shows that the quality of the linear approximation degrades as the contour becomes curved. In practice, this simple method works quite well near the curve. It is dependent on the variation of the normal directions of the curve hierarchy and tends to do less well as curves become more complex.

6.2 Guaranteed Error Bounds

We assume a linear approximation has been computed and an error bound ε_k must be produced. As will be shown in section 6.3 it is always possible to establish conservative bounds on the distance function with respect to the bounding regions of the curve

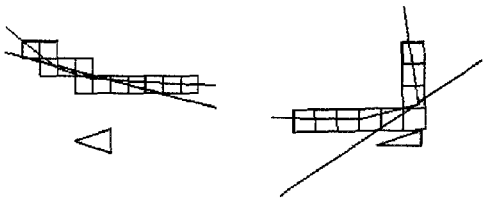


Fig. 9. Approximate zero sets of the distance function computed with averaged curve normals.

nodes. However, tighter bounds require knowledge of the behavior of the curve subset contained in the active list.

Figure 10 illustrates how an error bound ε_k constrains the location of the contributing curve points. A constant approximation has no estimate of the gradient and only constrains contributing curve points to the annulus centered at $\boldsymbol{x}$ denoted by the dashed circles. A linear approximation with a guaranteed error ε_k further restricts the possible locations of contributing points to the shaded area between the isocontours of the linear approximation at $\pm\varepsilon_k$.

Given a possible ε_k, the distribution of curve points in the active list must be analyzed to insure that at least one curve point falls in the shaded area for each point in T_k. Gaps or folds in the curve must be accounted for to insure that the bound on the distance is correct. The general procedure is as follows:

1. Compute a possible error bound ε_k.
2. Insure that the curve does not have folds or loops by requiring that the curve bend no more than 90 degrees from the estimated gradient $\boldsymbol{g}_k$ and is entirely front facing or entirely back facing with respect to $\boldsymbol{g}_k$.
3. Insure that no gaps exist in the curve by requiring that no curve nodes in the active list A_k labeled as gaps exist in the shaded area of figure 10.
4. If both 2 and 3 are satisfied, then the ε_k computed in 1 is a guaranteed bound.

6.3 Bound Culling And Constant Approximations

In this section we describe how conservative guaranteed error bounds are established and used to cull curve nodes and compute constant approximations. The algorithm relies on conservative lower and upper bounds on the unsigned distance within a triangle T_k with respect to the bounding regions of the curve nodes in the active list A_k. First, bounds on the distance function induced by each curve node are computed. Second, a bound on the distance function with respect to all nodes in the active list is computed. Finally, the individual curve node bounds are compared with the overall distance bounds. This comparison determines which curve nodes can not contribute to the distance. The following definition specifies when a point on the curve contributes to the distance function:

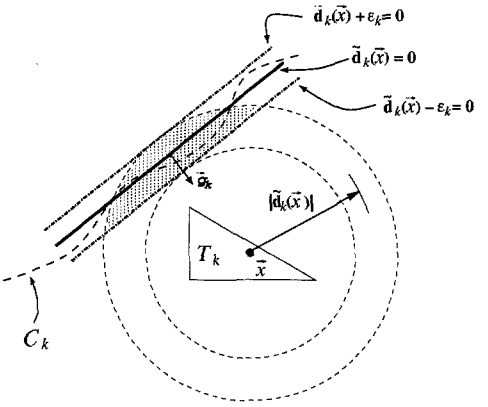


Fig. 10. Points of closest approach must fall inside the shaded area for a linear approximation with guaranteed error.

Definition 1. *A point $\boldsymbol{y} \in C$ contributes to the distance function within the triangle T_k of a distance node k if there exists at least one point $\boldsymbol{x} \in T_k$ for which $\boldsymbol{y}$ is the point of closest approach of the curve C.*

Furthermore, we say a curve hierarchy node α contributes to the distance function within T_k if at least one point $\boldsymbol{y} \in C \cap B_\alpha$ contributes.

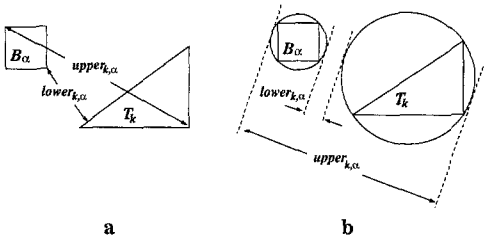


Fig. 11. Computing lower and upper distance bounds with respect to a curve hierarchy node.

We define conservative lower and upper bounds on the distance function with respect to a curve hierarchy node as follows:

$$lower_{k,\alpha} = \min(\| \boldsymbol{x} - \boldsymbol{y} \|) \tag{7}$$

$$upper_{k,\alpha} = \max(\| \boldsymbol{x} - \boldsymbol{y} \|) \tag{8}$$

for all $\boldsymbol{x} \in T_k$ and all $\boldsymbol{y} \in B_\alpha$. The lower and upper distance bounds are are conservative because they rely only on the bounding region B_α, and not on the actual curve contained within B_α. Figure 11a shows the lower and upper distance bounds for

an isocontour node and a distance node. These bounds can be expensive to compute and become more expensive in three dimensions. Efficiency can be increased by using bounding circles for both bounding regions as shown in figure 11b. The trade off is that bounds are more conservative and will not cull as many nodes from the active list.

The next step is to bound the distance function due to all curve nodes in the active list A_k of distance node k. These bounds are given by:

$$d_{lower} = \min(lower_{k,\alpha}) \; ; \; \forall \alpha \in A_k \tag{9}$$

$$d_{upper} = \min(upper_{k,\alpha}) \; ; \; \forall \alpha \in A_k \tag{10}$$

Thus, we have the following condition on the distance function within distance node k:

$$d_{lower} \leq \mathrm{d}(\boldsymbol{x}) \leq d_{upper} \; ; \; \forall \boldsymbol{x} \in T_k \tag{11}$$

Equations (9) and (10) define a piecewise constant approximation of the unsigned distance within distance node k.

A curve hierarchy node does not contribute to the distance function if the following condition holds:

$$lower_{k,\alpha} > d_{upper} \tag{12}$$

Finally, the lower and upper bounds may be used to construct a piecewise discontinuous constant approximation as follows:

$$\tilde{d}_k(\boldsymbol{x}) = \frac{1}{2}\mathrm{sign}(\boldsymbol{x})(d_{lower} + d_{upper}) \tag{13}$$

$$\varepsilon_k = |d_{upper} - \tilde{d}_k(\boldsymbol{x})| \tag{14}$$

7 Results

Fig. 12. Slice of a turbulent mixing simulation showing an isocontour.

We tested our algorithm on scaler data from a turbulent mixing simulation [6]. The simulation saved 270 time steps containing the entropy of each cell in the simulation.

Figure 12 shows a 256×128 slice from one zone of this computation with the isocontour at 50% entropy. The images and timings that follow are based on this isocontour.

The distance transform refines the triangle bintree until the error of the approximation satisfies a user defined criterion. The error criterion we used in the examples was computed as follows:

$$\text{uerror}(k, \varepsilon_{min}, \lambda) := \max(\varepsilon_{min},\ \lambda d_{lower}) \tag{15}$$

where d_{lower} is computed for each distance node k as in equation 9. This error criterion allows less accurate approximations farther away from the curve and clamps approximations near the curve to a user defined minimum.

Table 1 shows elapsed times for computing the distance approximation for various values of λ. Note that eventually the approximations near to the isocontour dominate the run times. However, the flexibility of focusing computation only where needed is an attractive feature of the algorithm.

ε_{min}	λ	Time(s)
0.5	0.5	57
0.5	1.0	42
0.5	10.0	26
0.5	20.0	25

Table 1. Elapsed times to compute approximations for various values of λ.

Figure 13 shows the distance approximations resulting from setting ε_{min} to 0.5 and λ to 0.5 and 10 respectively. The approximations far from the isocontour are much coarser in 13b.

8 Conclusion

We have implemented a fully adaptive distance transform algorithm that produces piecewise discontinuous linear approximations in a top down fashion. The algorithm is tuned for approximations near the input curve and tends to rely on constant approximations farther away. This could be imporoved by using a different approximation strategy for regions farther from the curve.

Although the implementation was not coded for speed, it is clear that a three dimensional version of the code would be quite slow. We believe that a hybrid approach might be succesful. The present algorithm would function as an outer loop on the distance nodes, while within each node a much faster approach could the used. For example, each distance node could be regularly sampled and a heuristic propagation scheme could be applied within each node.

9 Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. We wish to thank Valerio Pascucci for helpful discussions, and the staff at the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory for their advice and the use of their equipment.

References

1. Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scaler fields from 3d scans. In *Proceedings of SIGGRAPH*, pages 109–118, July 1995.
2. David E. Breen, Sean Mauch, and Ross T. Whitaker. 3d scan conversion of csg models into distance volumes. In *Proc. 1998 Symposium on Volume Visualization*, pages 7–14, Oct 1998.
3. H Edelsbrunner and E Mucke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13:43–72, Jan 1994.
4. Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptive sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of SIGGRAPH 2000*, pages 249–254, July 2000.
5. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuelzle. Surface reconstruction from unorganized points. In *Proceedings of SIGGRAPH 92*, pages 71–78, July 1992.
6. A. A. Mirin, R. H. Cohen, B. C. Curtis, W. P. Dannevik, A. M. Dimits, M. A. Duchaineau, D. E. Eliason, D. R. Schikore, S. E. Anderson, D. H. Porter, P. R. Woodward, L. J. Shieh, and S. W. White. Very high resolution simulation of compressible turbulence on the IBM-SP system. In ACM, editor, *Super Computing 1999, Oregon*, pages ??–?? ACM Press and IEEE Computer Society Press, 1999.
7. V. Pascucci and C. L. Bajaj. time-critical isosurface refinement and smoothing. In *Proceedings of Volume Visualization and Graphics Symposium 2000*. IEEE and ACM/SIGGRAPH.
8. J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Science*, 93:1591–1595, 1996.
9. Jane Wilhelms and Allen Van Gelder. Octree for faster isosurface generation. *Transactions on Graphics*, 11(3):210–227, July 1992.

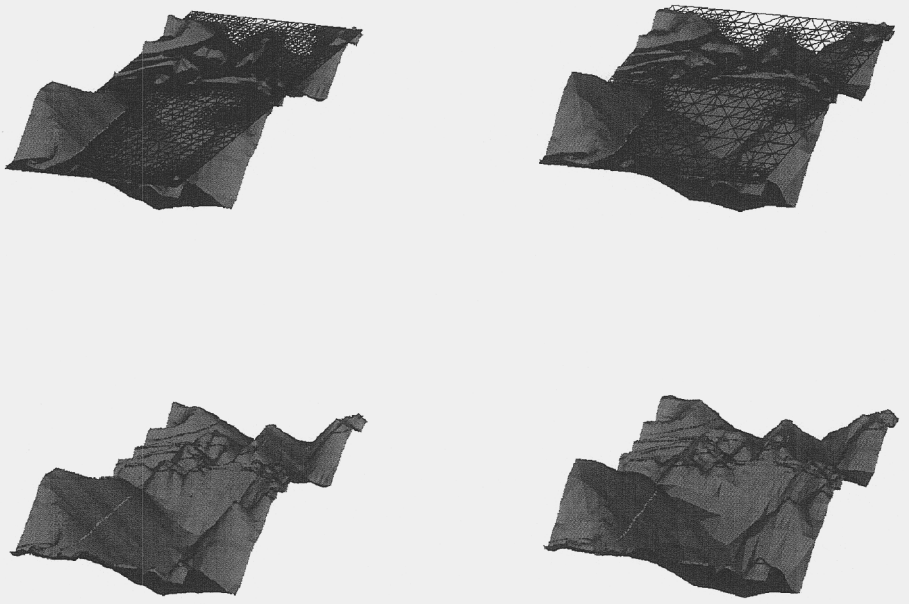


Fig. 13. Distance approximations for two values of λ.